TITLE:

数値計算と入力設計 (数学的ソフトウェアの評価)

AUTHOR(S):

木村, 泉

CITATION:

木村, 泉. 数値計算と入力設計 (数学的ソフトウェアの評価). 数理解析研究所講究録 1979, 359: 125-151

ISSUE DATE:

1979-07

URL:

http://hdl.handle.net/2433/104509

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
359巻 1979年 125-151

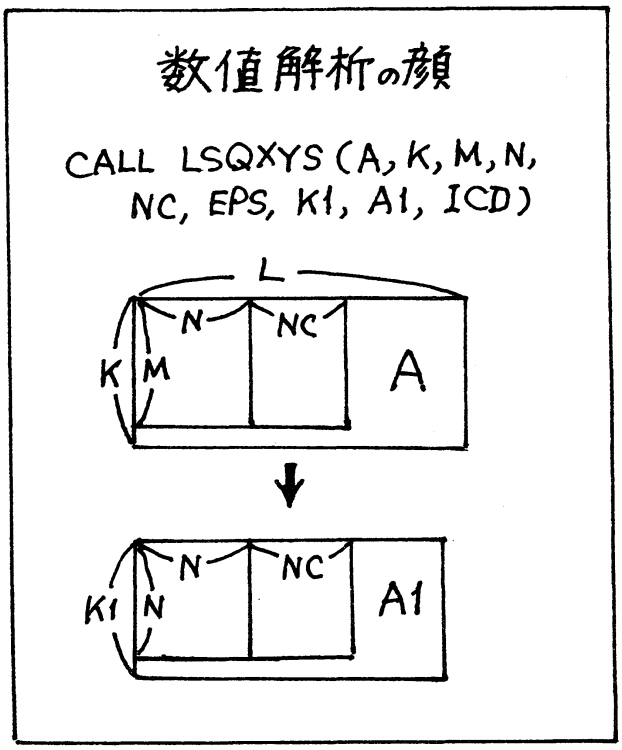
数値計算と入力設計

東工大 理　木村 泉

1

筆者は現在、プログラミングの方法、ソフトウェアエンジニアリングといった方面を主たる興味分野としており、数値計算そのものについては門外漢である。しかし、上記のような立場から見た数値計算（ないし数値解析）には大いに興味をもっているので、そういう者としての考えの一端をここにご被露させていただきたい。

さて、「数値解析」ということばをきいたときに、筆者がまっさきに、（少なからざる懼れの気持とともに）思い浮べるのは、数値計算用ライブラリプログラムの説明書の随所にあらわれる長い長い呼び出し列である。それは、たとえばスライド1に示すような「顔」をしている。筆者はかねてから自分がうっかり者であることについて絶対的な確信をいだいており、こういうものを使おうとしたら少なくとも3度に1



スライド1

度はまちがえるだろう、自分は数値解析を勉強する資格がないのではないか、と心淋しく思う。*

---

*この例題は、たまたま手近にあった科学計算用サブルーチンライブラリのマニュアルから借用したが、この特定の設計を批判する意図は毛頭ないので、変数名その他をわざと変えて出所が明確になりすぎないようにしてある。筆者の知るかぎりにおいて、数値計算用のライブラリはほとんどすべてこういうスタイルをとっており、またこのスタイルに相当の必然性があることは重々承知の上である。

この呼び出し列は、単精度最小２乗解計算用サブルーチンLSQXYSを呼び出すものである。（CALL文が2行にまたがっているのはスペースの都合による。本来は1行に書くべきものである。）M行(N+NC)列の行列Aを与えられて、対応する最小２乗解を、N行(N+NC)列の行列A1の中に作り出す。Aの頭N列が係数行列（M行N列）を与え、うしろNC列は、NC個の定数列ベクトル（長さM）を与える。LSQXYSは、係数行列を共有するNC個の別々の最小2乗問題をまとめて解くはたらきをする。

A1のうしろNC列には、NC個の列ベクトル（長さN）として、求める最小２乗解が入る。なおA1は、LSQXYSによって、作業場所としても利用される。

配列A、A1は利用者が用意する。その際、AをK行L列の配列に、またA1をK1行L1列の配列に、はめ込んで渡すことができるようになっている。したがって、A、A1の入れ場所としては、ありあわせのあき行列を利用することができる。EPSはピボット要素が小さくなりすぎた際の打ち切り条件を与える実数、ICDは実行が正常に終了したかどうか、異常終了したときはその原因は何であるかを示す整数型の変数である。実行終了時のICDの内容を、スライド2にまとめておく。

誤りの表示

ICD：整数型の変数名

- 0　　… 正常
- 10000 … 入力パラメタの誤り、
  すなわち、
  K < M,
  M < N,
  N ≤ 0,　　または
  K1 < N　　のとき。* **
- 上記以外 … |ピボット| < EPS で
  打ち切ったとき、そのときまでの
  消去回数

---

* EPS > 0 は確認しない。(EPS ≤ 0 を
積極的に使うつもり?)
** L ≥ N+NC, L1 ≥ N+NC はノーチェック.

スライド2

以上は、この種のライブラリサブルーチンの設計としてはきわめて常識的であるが、ソフトウェア工学の立場に立って潔癖に検討してみると、スライド3に示すような問題点が浮び上ってくる。念のためにもう少し補足すれば、次の通りである。

オ1に、スライド1の呼び出し列は何しろ長すぎる。よほど一生懸命やらないと、順序をまちがえたり、一つ抜かしたりしかねない。そして、そういうとき起ることは、きわめて

文　句
1. そんなにぞろぞろ書かされたら、おぼえきれない。
2. A1を用意させられるのはいやな感じ。
3. KやK1は問題の筋道に関係ない。
4. K、K1があってL、L1がないのはご都合主義。
5. ICDの中味が自明でない。
6. EPSは傷口。
7. LSQXYSとは変な名前。

スライド3

悲劇的である。

オ2に、A1は出力が入るところなのだから、それを利用者に用意させるのは、筋違いといえば筋違いである。ライブラリの方で適当に見つくろって用意してくれ、といってもバチは当らないはずである。またオ3に、K、K1は記憶場所の節約に関係した引数で、最小2乗解という問題の真の筋道とは関係がない。そういうものについて利用者に心配をかけるのは、これも筋違いといえば筋違いである。

オ4に、埋め込み場所をなす配列A、A1の行数K、K1が与えてあって列数L、L1が与えてないのは、Fortran

の文法上、なしでもすむからだが、ご都合主義のそしりをまぬかれない。これなしでもDOループは書けるかも知れないが、L、L1が小さすぎればどんなひどいことでも立派に起こるので、そうでないことが呼び出し時点で確認されない（スライド2）のは問題である。また才5に、変数ICDの中に返される値は、正常終了のとき0、という一点を別とすれば明白性（understandability）に欠けている。10000という数は「きり」のよい大きな数、という以外の必然的根拠をもたないのでおぼえにくいし、また実は、ここのところが社によって少しずつちがっているために機種間の移植の際にこまったことが起りやすいことはよく知られている。

才6に、EPSなどというものがここに出てくるのは、あえてやや無理なことをいえば、数値解析が100%発達しつくしていないために生じた傷口である。利用者は最小2乗解が求めたいのであって、最小2乗解計算用のある特定のアルゴリズムを実行したいのではない。理想をいえば、問題そのものでなく解法の方にだけ関係するEPSのようなものは、表には出てこないことが望ましい。

そして最後に、LSQXYSという名前にも、やかましくいえば問題がある。この名前を一々マニュアルをひもといてみることなしに自信をもって使いこなすことはむずかしい。こ

こは、理想をいえば漢字かなまじりの名前を使って

CALL 単精度最小2乗解(.....)

と書きたいところである。また、ナマにこう書くのは無理としても、'LSQXYS'の'XY'はじゃまである。こういうふうになっているのは、いろいろな単精度最小2乗解サブルーチンが出てきたとき、少しずつ名前を変えて使いわけがしたくなるかも知れないから、ということと、利用者プログラムの方で使っている外部名となるべく衝突したくないから、ということの二つがあるためと思われるが、前の方の問題点については、こういう余計な配慮をすると古いサブルーチンを新版にさしかえるときに呼び出し側を変更しなければならなくなって、かえってよくないこともありうるし、あとの方の問題点については、これは要するにFortranが悪いのである。システムの外部名と利用者の外部名が必要に応じて切りわけられるようになっていれば、こんな無理な配慮はいらないはずである。

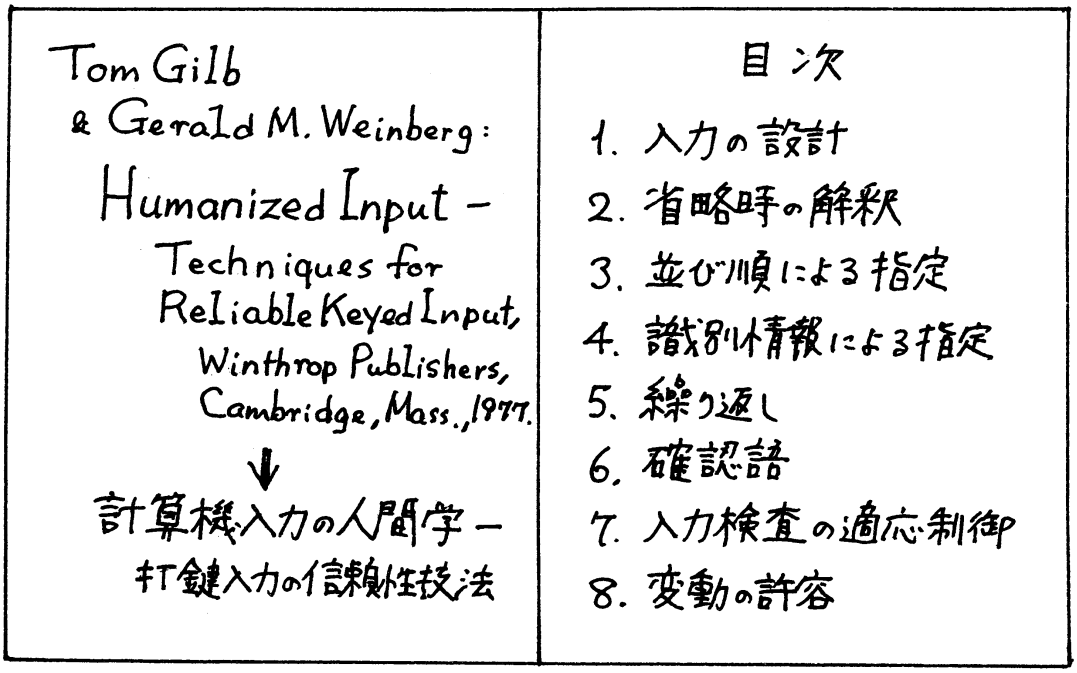
2

この種の話は、誰しもがすっきりと納得するような形で一刀両断に議論することがむずかしい。「好みの問題だ」、ときめつけられてしまうとそれっきりのようなところがある。

そのむずかしさの源泉は、人間というものが入り込んできている、というところにある。早い話、スライド3の第1項で「たくさん書かされたらおぼえきれない」のは人間の記憶力に限界があるからである。また「AIを用意させられるのはいや」というが、それは用意させられる主体が人間だからこまるのであって、計算機の方で用意してくれるのなら「どうぞご随意に」となるところである。そもそも人間というものは、途方もなくとらえどころのないものであるから、そういう人間にからむ話が漠然としたことになりやすいのは当然の成り行きである。

そんな漠然とした話は非科学的だからよそう、といってしまえるなら簡単だが、実は数値計算用ソフトウェアの信頼性を論じようとすれば、人間的要素が入り込んでくることは避けられない。というのは、信頼性とはつきつめていうと使う人間にとって安心できるか、という問題だからである。

そういう漠然とした問題を取り扱うにはどうしたらよいだろうか。よく考えて、さしあたりいえることをはっきりいいあらわす、というほかあるまい。たとえば注意事項を箇条書きしてみる、というのは、いかにも大ざっぱのようだがこの場では有力な手助けになる。箇条書きは「精密さ」とはほど遠い存在であるかも知れないが、箇条書きがある状態は箇条



スライド4　　スライド5

書きすら存在しない混乱状態と比べれば、どれほどよいかわからない。

この種の箇条書き工学をまじめにやってみせた例として、スライド4に示す本は近来のヒットと思う。（矢印の下は、表題を和訳すればこうなる、ということを示す。）計算機入力に関するさまざまな問題点を、スライド5に示すような8個の章にまとめ、豊富な実例を添えて勇弁に論じている。主題は、オペレータが打ち込むデータをどのような形式にしたら信頼性が高まるか、というところにあるが、そこに書かれていることの多くは数値計算用ソフトウェアの設計にも、直接、間接に参考となる。

たとえば、スライド3の第1項に関係する話として、こんな話が出ている。項目をずらずら並べて、その何番目に出てきたかによって各項目にそれぞれの意味をもたせたものをスライド4の本では並び順による指定（positional messages）と呼んでいるが、その項目数はふつうは3-4項目が限度だという。その一応の説明として著者らは、こんな議論をしている（原著77ページ）。

もし項目が2個であれば、まちがった順序は1通りしかない。項目が3個になると、まちがった順序は5通りあるが、それはまだ、どうこういうほど多くはない。項目が4個であれば、誤りは23種となる。それもまあがまんできる。項目が5個もあったら、誤りの種類は119個にものぼる。4項目というのはぎりぎりの限度だろう。——この議論を承認するなら、スライド1において引数が9個もあるのは圧倒的に多すぎるということになる。なぜなら、Fortranのサブルーチン呼び出しにおける引数の列は、まさに一種の並び順指定だからである。

ではどうしたらよいか？　スライド4の本の著者らは、一つの逃げ道として、入力データに構造をもたせるという手があることを指摘している。（このほか、順序がどうでもよい場合、および計算機の中で並べなおせる場合は項目が多くて

もよい、とされている。）たとえば

PROF. SIN HITOTUMATU 075 751 2111

とあったとき、われわれはこれを「6項目もあるから誤りの可能性が719通りもあるではないか。」というふうには感じない。PROF. SIN HITOTUMATU を一つの集団、075 751 2111 をもう一つの集団、というように感じる。それを入れちがえる可能性は1通りしかない。たとえば

2111 HITOTUMATU 075 PROF. 751 SIN

というような入力誤りが生じるとはまず考えられない。このような並びかたは719通りの中から除いていいわけである。

そういう逃げ道はないか、とスライド1をながめなおしてみると、大幅な改善はFortran のままではなかなかむずかしいようだが、小さな改善の余地は二、三目につく。一番目立つのは出力行列（計算結果のはいり場所）に関する引数がK1、A1の順、つまり行数、配列名の順で並んでいることである。これに対し、入力行列に関する引数はA、K、M、N、NCの順に並んでおり、配列名、行数は配列名の方が先に来ている。こういう不均一性はまちがいのもとである。K1、A1はむしろA1、K1の順に並べなおすべきであろう。

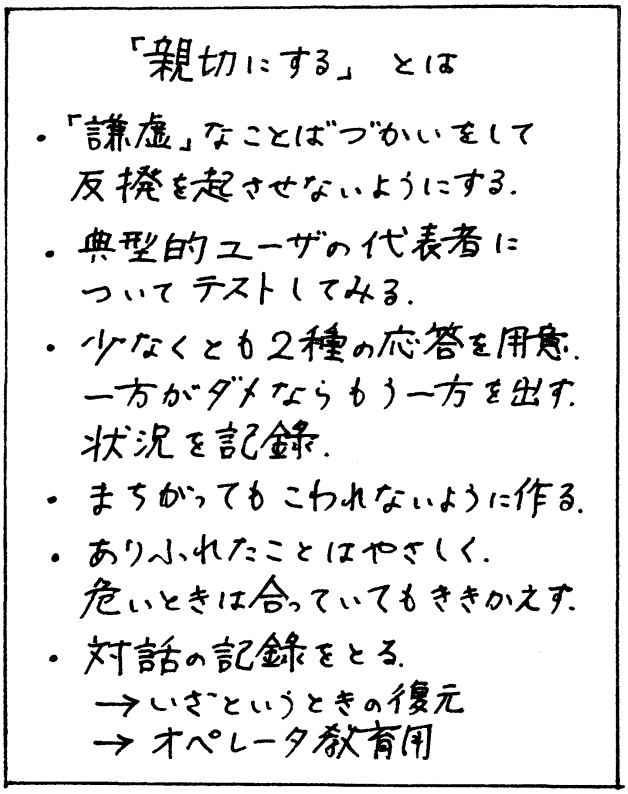
そこでそう並べなおしてみると、今度はA、KとA1、K1の間にM、N、NC、EPSがはさまっているのが気になってくる。というのは、今度はA、KとA1、K1がいわば対句を形造るので、その対称性を途中に長い列をはさむことによってこわすのは好ましくないからである。そこで、A、Kなどは物理的な入れ場所に関する事項、M、N、NCは解くべき問題そのものに関する事項、というふうに切りわけて考え、呼び出し列を

CALL LSQXYS(A, K, A1, K1, M, N, NC, EPS, ICD)

となおせば、まだまだ不満ながらいくらかすっきりするように思う。

そのほか、たとえばスライド3の問題点5についても、スライド4の本にはきくべき忠告が含まれている。すなわち第8章（「変動の許容」の章）には、データ入力システムが利用者に対して親切であることの「定義」として、スライド6のようなものが与えられている。これはあいてが端末の前にすわっているとしての話だから、スライド3とは状況がちがうともいえるが、誤りの処理に関する一般的な心がまえ、という面ではたいへん参考になる。

たとえばスライド6の第3項目には、少なくとも2種の応答を用意せよ、というものがあがっている。利用者というも



スライド6

のは、計算機から出されたメッセージを、いつもいつもさっと理解できるとは限らない。だから利用者には、「え?」ときかえせるようにしてやらなければいけない。ききかえされたら前とは視点を変えた説明を与えるべきだ。そして、どのメッセージに対して利用者がききかえしたかを記録し、後日改善の資料とせよ、というのである。

今の場合、利用者が直接サブルーチンLSQXYSにききかえす、というのは平仄の合わない話だが、LSQXYSを組み込んだシステムの利用者にききかえす権利を与えよ、というのはまさにその通りだし、そしてLSQXYSを使ってシステ

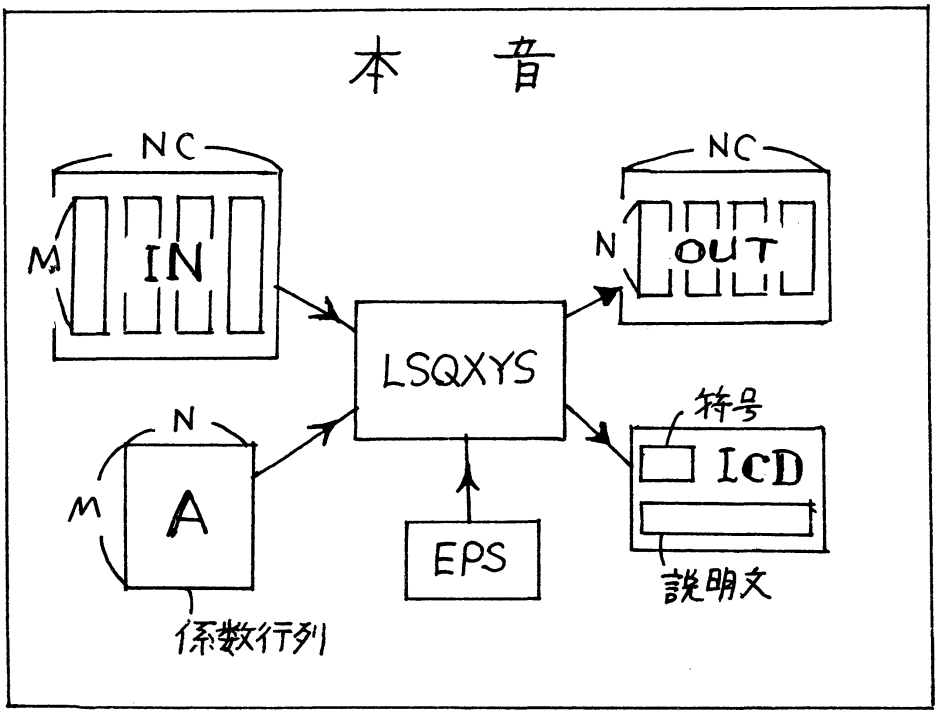
ムを作る人がそういうことをしやすいようにしてやりなさいというのも、たいせつな話である。そう考えてみると、ICDの中に10000というような整数が入るだけ、というのは、やはりぶっきら棒のそしりをまぬかれない。誤りがあるかないか、あったとすればそれは大別してどういう種類の誤りか、といったことがさっとわかるように、たとえば0、1、-1といった、はっきりした値を返す一方、それとは別にもっと人間向きの説明文も出てくることが望ましい。「Nは定数行列の列数だから負ではいけない。」とか「残念ながら才....回目の繰り返しで |枢軸要素| < EPS = ... 。」とかいうような具体的な説明文が組み込みになっていて、容易に取り出して使える、ということであるべきだ。こういうことはLSQXYS自身が一番よく「知っている」はずのことだから、LSQXYSの方で用意するのが自然である。サブルーチンを使ってシステム作りをする人の方が外側の問題をよく知っていることは事実だが、上記のようなきめこまかい説明文が内側で用意されていてはじめて、それを加工することにより、楽に、問題を知っているという有利な立場を生かすことができるのであって、そういう下ごしらえがないとシステム作成者としてもついめんどうになり、手を抜きたくなるおそれがある。

スライド4の本の内容は多岐にわたるので、そのすべてをここで紹介することは不可能であるが、そのつもりになって読んで行くとほとんどいたるところに数学ソフトウェアのパッケージング（まとめかた）に関係したアイディアがみつかる。数値計算をご専門のかたがたにも、ご一読をおすすめする。

3

こういう人間がらみの問題が注目されるようになってきたのは、数値計算に限らず、広く計算機応用一般にわたって比較的最近のことである。かつては計算機資源がたいへん高価についていたために、付随する人的コストはそれと比べればものの数ではなかった。したがって、前節で紹介されたような話がまじめに議論されることは至って少なかった。その時代の記憶があまりにもあざやかであるためにとかく思いちがいをしやすいのだが、状況は根本的に変ってしまっている。数値計算に関しても、今までようはずっとまじめに、人的問題をとりあげてみるべき時期である、と思う。この新時代にわれわれはいかに対処すべきであろうか？

与えられた計算機システムをどう使いこなそうかと苦心する前に、まず本音をいいたいだけいってみて、その上でその



スライド7

本音にできるだけ近いものを実現するにはどうしたらよいかと考える、というのが最近のソフトウェア工学の一つの基本的な流れである。そこで、スライド1の呼び出し列によってわれわれは結局のところ何をやりたいのか、われわれの本音は何か、と考えてみると、それはスライド7のようなことであるにちがいない、と気づく。

すなわち、スライド1で係数行列と定数行列を一つの配列につめ込んで渡しているのは便宜上のことで、本音ではそれらを別々のものとして渡したいのである。係数行列は一応正真正銘の2次元配列であるが、スライド1でのAのうしろ

NC列は実はNC個の列ベクトルの集合体であって、2次元の行列でなくむしろ1次元列ベクトルを成分とする1次元ベクトルと考えた方がすっきりする。また、行列の大きさに関するパラメタN, M, NCは、本来は行列の属性の一部であるはずだから、配列とこれらのパラメタを別々にサブルーチンに引き渡すかわりに、これらのパラメタを内部に含む行列というものを引き渡す、と考えた方が筋が通る。誤りの表示ICDも、利用者の立場に応じて0、1、−1のような単純明快な符号と詳細な説明文のどちらでも取り出せるようにしたい。そしてそのためには、ICDは単なる整数型の変数値でなく、上記2種のものの組にしておくことが望ましい。*

---

*実はこれでも遠慮しているのである。NC個の列ベクトルを与えているのは、ほんとうは列ベクトルを1個ずつ与えてNC回LSQXYSを呼び出したいものを能率向上のためにこうしているものだし、EPS、ICDも、できれば、ピボットの値が非常識に小さいとき、入力パラメタがおかしいときなどに限って正体をあらわして利用者に意見を求め、ふだんは存在を意識されないようなものであってほしい。そこまであらわに書いてはあまりにドラスティックだから、一応上のようなことで手を打ってある。

このように書き出してみると、われわれの本音をFortranのわく（ないしはFortranの旧来の、常識的なわく）に押し込めようとすることがどんなに無理なことか、よくわかってくる。たとえばFortranには、行数と列数と要素配列をひとまとめにして、行列という抽象的な概念を作り出す機能など、表立っては用意されていない。ICDを符号と説明文の組と見ることがむずかしいことも同様である。

本来なら、スライド7に対応して、せめてスライド8のようなことが書きたいのである。ここで、[(a)].[(b)]は、構造物(a)の構成要素(b)という意味をもつ。これなら

せめて...

OUT = LSQXYS(A, IN, EPS, ICD)

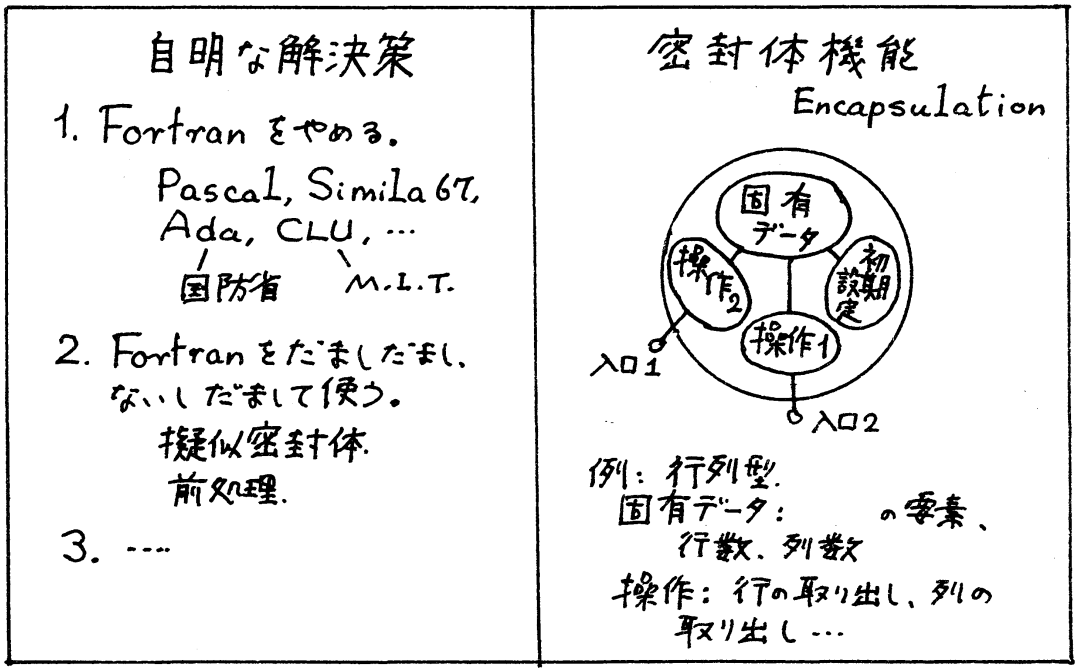
ただし

A.M	は 定数行列 の列数、
A.N	は 〃 の行数、
A.ELEM[i,j]	は 〃 の i,j 要素.
IN.NC	は 入力列ベクトルの個数、
IN.M	は 〃 の長さ、
IN.V[i]	は 第i番の入力列ベクトル.
OUT....	は IN.... とほぼ同様.
ICD.CODE	は 0、1、-1、...、
ICD.MESSAGE	は 説明文.

スライド8

呼び出し列の長さは、かろうじてスライド4の本の規準に合っている。その他、スライド3の文句のかなりの部分に答えたことになっているが、むろん残念ながら旧来のFortranではこんなことは書けない。

数値解析専門家のかたがたにわかにご賛成いただけるかどうかはわからないが、ソフトウェア工学屋にとっての一つの自明な解決策は、言語を変えることである（スライド9）。たとえばPascal は、新しい型を導入する機能をもっている。また、Simula 67、およびその影響を受けたAda CLU などのような近来の多くの言語では、さらに進んで、スライド10に示すような密封体機能（encapsulation）

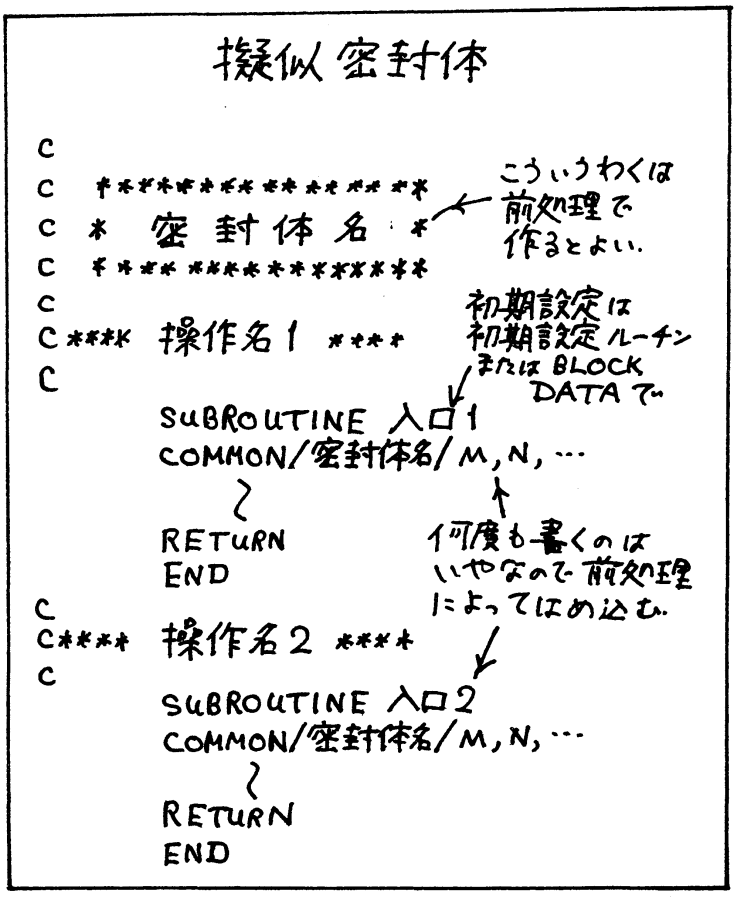


スライド9　　スライド10

を含んでいる。Adaは米国国防省が制定準備中の新言語、CLUはM.I.T.で開発された言語であるが、これらは単なる例であり、密封体機能（何らかの形の）をそなえた言語は数多くある。

ここで密封体機能とは、固有の共通データとそれをいじるための操作手続きをひとまとめにして取り扱う機能である。たとえば行列の要素と行数と列数を固有データとし、行を取り出す操作、列を取り出す操作などを操作手続きとして用意し、これによって行列型といったものを作り出すようなことが考えられる。操作手続きを介さないで、じかに固有データをいじることは、しない建前、ないしできない建前である。この機能を使えば、スライド8の理想は実現できる。

このほか、比較的受け入れられやすいであろうと思われる方向として、Fortranをだましだまし、ないしだまして使う、というものもある（スライド9、オ2項）。たとえばスライド11に示すように、名前付き共通領域を中心として、操作用サブルーチンを配置し、これらを注釈でふちどる、というやりかたで、密封体風の気分を出すことも考えられる。書くことが多くなって、めんどうではあるが、その問題はプログラムの前処理によってある程度解決できる。前処理プログラムをがんばれば、スライド8に近いものから出発するこ



スライド11

とも不可能とではない。

また一方、スライド9のオ1項をいっそう押し進めて、たとえばAPLのようなものを考慮することもできよう。いずれにせよ、Fortranを使い続ける、という以外の発想はないものか、一度しっかり考えてみる必要があるのではないだろうか?

4

こういう提言に対しては、そんな効率のわるいことなどとてもできないよ、という趣旨のご反対もあろうかと思う。そこで、計算の効率ということについて、ここで少し考えてみることにする。

スライド8では、係数行列Aと列ベクトル群INが別々に与えられている。これはその方が問題の本音（スライド7）に近いからであり、やろうと思えばFortranでもやれることだが、多くの数値計算専門家はこれをとんでもない非能率とお感じになると思う。たしかにそれは、効率に関する伝統的な感じかたである。このようにすると、本番の計算をはじめる前に、AとINを組み合わせてひとまとまりの横長の行列を作る作業をしなければならず、そのために何ほどかの計算時間を食われることになる。

だが、よく考えてみるとその作業は、最初に1回おこなわれるコピー作業にすぎない。スライド1の場合、最初にデータを読み込むときからM行（N+NC）列の様式を頭において適当なREAD文を使うようなことをすれば、そのコピー作業をなしですますこともできるわけだが、そうやって達成される節約は本番の計算、ないしREAD文の所要時間と比べ

れば、無視できる程度に小さいのがふつうである。*

実は、そこまで効率を気にするのであれば、そもそもサブルーチンなど使っていたのではダメな理屈である。サブルーチンの引数の受け渡しには何ほどかの計算時間が消費される。機種、処理系によって様子はちがうが、その計算時間消費量が上記のコピー作業による消費時間を超えていたとしても驚くにはあたらない。サブルーチンを呼ぶのはやめて、本体を呼び出し側に取り込んでしまった方がよい、ということになるのかも知れない。

もちろんそんなことをする人はだれもいない。スライド1のようにする方が、まだしも思考の節約になり、まともなプログラムが早くできるからである。

ずらずら主プログラムに書き込むかわりにスライド1のようにサブルーチンを呼び出す、というのと、スライド1のように係数行列と列ベクトル群をいっしょにしないでスライド8のように区別して扱う、というのは、実は方向としては同じである。スライド1のようなものがはじめて使われたころから比べれば、計算時間の値段はずいぶん安くなっている。

---

*Kernighan & Plauger: The Elements of Programming Style (木村訳、プログラム書法)、問題1.1.

これからもずっとスライド1のところで立ち止っていなければならない理由はないと思う。

また処理系の作りかたいかんによっては、スライド8の方が本音があらわに出ているだけに、いっそう高級な最適化（たとえばサブルーチン本体を呼び出し側に自動的に転写すること——インライン展開——など）ができてかえってもうかる、という可能性すらある。

1960年代のはじめごろ、計算機関係の広告に、“our customers' exacting needs” という宣伝文句をよく見掛けたように思う。当社ではお客様のきびしいご注文に沿って、最小の機器構成で最大の効率をあげるよう努力しております、ということだったのだと思う。（Communications of the ACM あたりかと思って探してみたが、どうも見当らない。）

いずれにせよ、この標語が想起させるような、計算機の能力を100%発揮させなければならなかった時代は、もうとうに終っている。そうするために要する人力、またはそのために生ずる危険があまりに大きいのであれば、その度合に応じてたとえば能力の90%とか85%とかで満足することにし、生じた余裕を利用者の身体を楽にすること、安全性を高めることに利用したとしても、今やかしも恥じるには及ばな

い時代である。そういうバランス感覚に立脚した、冷静な態度こそ、今もっとも求められているものであると思う。これをもってこの話の一応の結論としたい。

討　論

Q. スライド1のK, K1のようなものは領域共用のために使われているのだが、これは数値計算用ライブラリの方で自前で領域管理をするつもりならなしですむものである。すなわち、ライブラリをバラバラなサブルーチンの集うとみるのでなく、一つのまとまったシステムとして扱うつもりなら回避できるものであって、そこのところは必ずしもFortranの罪とはいえないのではないか？

A. そういう管理をFortranの範囲内でやろうとすると、そのための管理ルーチンや共通領域が必要になる。そうすると、それらのルーチンや領域の名前が利用者側で使っているものと抵触するおそれが出てくる。これまで領域の自動管理がおこなわれていなかったのは、そういう迷惑を利用者にかけまいとする、ライブラリ作成者の良心的な態度によるところが大きいと思うが、それはさかのぼっていえばFortranにおいて、この種の名前に階層がないことに起因する問題であり、Fortranもまた罪なしとはいえな

い。

Q. 近ごろSimula 67を数値計算に使用してみている。違いのはこまるが、概念がわりあいすなおに表現できて具合がよいように思っている。このことについてどう思うか?

A. わが意を得た、という感じであり、討論者に敬意を表する。違い点についてはSimula 67の特殊性によるところもあると思う。

Q. UMSをどう思うか?

A. UMSの、Fortranに前処理によってMATRIX型、VECTOR型などを導入しようという方向は、たしかに正しいが、具体的な設計の上でUMSはFortranに気がねしすぎているような気がする。

Q. スライド1のようなものをいやだ、と感ずるのは熱意がたりないからではないか? たしかに数値計算用ライブラリを使ってシステムを組み上げるのは大変な仕事である。特に引数の順序については誤りが起りやすく、非常に神経を使う。だがそうやって神経を使って一度まとめあげたものは気持よく使える。ああいうものは使いにくいからいやだ、などと口走るけしからぬ学生を甘やかすのは教育上おもしろくないのではないか?

A. スライド1のようなものを見たときどう感じるかについ

ては、たしかにコミットメントの程度いかんによる差があるだろうと思う。図目八目ということでおきいただきたい。

ソフトウェアの製造方法についてあれこれ考えている筆者のような者にとっては、まとまったシステムを作りあげるまでが大問題で、そこが大変だとすればやはり何とかしなければいけない、と感ずる。生産性も問題だが、一生懸命やってもなお残る誤りがこわい。

教育に関していえば、「もっとまじめにやれ」という可能性はつねに存在する。だが、無理なことを「まじめにやれ」といって学生に押しつけると、そこは何とかなっても他のところで学習能率がかたんと下るのは、経験の示すところである。これもまたバランスの問題であり、今の場合ライブラリの方をなおすのが筋だと思う。